NASA Technical Memorandum 112197

Neural Network Prediction of New Aircraft Design Coefficients

Magnus Nørgaard, Charles C. Jorgensen,
James C. Ross

May 1997

National Aeronautics and
Space Administration

NASA Technical Memorandum 112197

Neural Network Prediction of New Aircraft Design Coefficients

Magnus Nφrgaard, Institute of Automation, Technical University of Denmark
Charles C. Jorgensen, Ames Research Center, Moffett Field, California
James C. Ross, Ames Research Center, Moffett Field, California

May 1997

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

NEURAL NETWORK PREDICTION OF NEW AIRCRAFT DESIGN COEFFICIENTS

Magnus Nørgaard,* Charles C. Jorgensen, and James C. Ross
Computational Sciences Division
Ames Research Center

SUMMARY

This paper discusses a neural network tool for more effective aircraft design evaluations during wind tunnel tests. Using a hybrid neural network optimization method, we have produced fast and reliable predictions of aerodynamical coefficients, found optimal flap settings, and flap schedules. For validation, the tool was tested on a 55% scale model of the USAF/NASA Subsonic High Alpha Research Concept (SHARC) aircraft. Four different networks were trained to predict coefficients of lift, drag, moment of inertia, and lift drag ratio (C_L, C_D, C_M and L/D) from angle of attack and flap settings. The latter network was then used to determine an overall optimal flap setting and for finding optimal flap schedules.

INTRODUCTION

Wind tunnel testing can be slow and costly due to high personnel overhead and intensive power utilization. Thus, a method that reduces the time spent in a wind tunnel, as well as the work load associated with a test, are of major interest to airframe manufacturers and design engineers. Modern wind tunnels have become highly sophisticated test facilities used to measure a number of performance features of aircraft designs. In this study we have chosen to consider only determination of the coefficient of lift (C_L), coefficient of drag (C_D), pitching moment (C_M) and lift/drag ratio (L/D) as functions of angle of attack and flap settings. In this paper we emphasize prediction, but the techniques are applicable to other steps of wind tunnel testing as well.

Currently, a new design test is followed by extensive manual data fitting and analysis. To allow researchers to interpolate between measurements, evaluation of the aircraft design is based on visual inspection of curves. One way to automate the procedure is to find mathematical expressions to describe the complex relationships between variables. Although neural networks are not the only approach potentially able to perform this task, e.g., numerical aerodynamic simulations, such soft computing methods provide a very cost effective approach. Spin-off benefits can also result from a new approach and include increased automation of measurement processing and aids for checking earlier calculations. The longer term benefits are a significant reduction in costs and faster

---

* This author was at the NASA Ames Neuro-Engineering Laboratory in 1994 as part of a cooperative student work study program between NASA and the Institute of Automation, Electronics Institute, and Institute of Mathematical Modelling, at the Technical University of Denmark. The Danish Research Council is gratefully acknowledged for providing financial support during his stay.

development of new aircraft, or alternate tunnel uses such as more aerodynamically efficient automotive design.

This paper is organized as follows: A short introduction to Multilayer Perceptrons (MLP) is given and one powerful method we used (a variation on the Levenberg-Marquardt method) is presented. Next, we describe how a subset of test measurements were used with the technique to train four networks to predict aerodynamical coefficients and the L/D ratio, given angle of attack and flap settings. We then present two applications. The first addresses the problem of determining an "overall optimal" flap setting using a method based on integration of L/D vs. C_L. The second demonstrates an easy strategy to find optimal flap schedules. Finally, details of the software tool set are given in an appendix as a supplement to documentation in the project code.

MULTILAYER PERCEPTRONS

The phrase "neural network" is an umbrella covering a broad variety of different techniques. The most commercially used network type is probably the MLP network. See reference 1. An example of a MLP network is shown in figure 1. In this study we used a two-layer network with tangent hyperbolic activation functions in hidden layer units, and a linear transfer function in the output units. A two-layer network is not always an optimal choice of architecture (goodness measured in terms of the smallest number of weights required to obtain a given precision), but it is sufficient to approximate any continuous function arbitrary well (ref. 2), and training is easier to implement and faster in this case.

A MLP network is a special type of an "all-purpose" function, which in many recent situations has shown an excellent ability for function approximation (ref. 3). The network shown in figure 1 corresponds to the following functional form

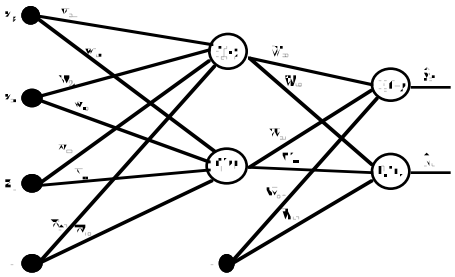


Figure 1. A three input, two output, two layer MLP network. The weights from the inputs set to 1 represent the biases. Here $f_i(x) = \tanh(x)$ and $F_j(x) = x$.

$$\hat{y}_i(w,W) = Fi\left(\sum_{j=1}^{q} W_{ij} h_j(w) + W_{i0}\right) = F_i\left(\sum_{j=1}^{q} W_{ij} f_j\left(\sum_{l=1}^{m} w_{jl} z_l + w_{j0}\right) + W_{i0}\right) \qquad (1)$$

A special feature offered by this type of network is that it can be trained to approximate many functions well, without requiring an extravagant amount of parameters (weights). This is discussed in Sjöberg, et.al., (ref. 4). One disadvantage compared to other network types is that training is slow

because the network implements a non-linear regression, i.e., there is a nonlinear relation between the adjustable parameters, the weights, and the output.

In this study, we were interested in enhancing generic neural networks for wind tunnel test estimation. Obtaining network training data is very costly, e.g., $3,000 dollars per tunnel hour for the National Full-Scale Aerodynamic Complex at Ames Research Center. Consequently, only limited data sets were available and usually as byproducts of previously scheduled tests. The size of the data set imposes an upper limit on how many weights the networks should contain. In practice, since there is also uncertainty associated with the measurements, the number of data points must exceed the number of weights by a sufficiently large factor to ensure that good generalization may be achieved. Training time, on the other hand, is not of prime importance. Many arguments can be made in favor of some form of MLP networks as the right choice for the given problem. The real problem is obtaining extremely high accuracies critical for commercial viability.

TRAINING

The training phase is the process of determining network weights from a collected set of measurement data. The treatment of different aerodynamical coefficients is essentially identical, so we will consider a generic quantity 'y' instead. If the aircraft flaps are coupled, y becomes a function of three different variables: Angle of attack (α), leading edge flap angle (LE), and trailing edge flap angle (TE).

$$y = g_0(\varphi) \tag{2}$$

where

$$\varphi = \begin{bmatrix} \alpha & LE & TE \end{bmatrix}^T \tag{3}$$

The function 'g' is unknown, but the wind tunnel tests provide us with a set of corresponding $y - \varphi$ pairs

$$Z^N = \left\{ \left[\varphi^i, y^i \right]; i = 1, \ldots, N \right\} \tag{4}$$

Naturally the measurements of y are not exact, but will be influenced in undesired ways from a number of different sources. All measurement errors are grouped in one additive noise term, e

$$y = g_0(\varphi) + e \tag{5}$$

The objective is now to train the neural network to predict y from

$$\hat{y} = \hat{g}(\varphi) \tag{6}$$

The predictor is found from the set of measurements, Z^N, from here on denoted the *training set.* Expressed precisely, we wish to determine a mapping from the set of measurements to the set of functions contained in the chosen network architecture $g(\varphi; \theta)$

$$Z^N \rightarrow \hat{\theta} \tag{7}$$

so that $\hat{y}$ is *close* to the "true" y. θ is the parameter vector containing all adjustable parameters (in this case, the network weights).

A common definition for goodness of fit in neural nets is the mean square error

$$V(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y^i - \hat{y}^i(\theta))^2 = \frac{1}{2N}\sum_{i=1}^{N}(\varepsilon^i(\theta))^2 \tag{8}$$

Thus, training becomes a conventional unconstrained optimization problem. For various reasons back-propagation, a flexible but somewhat ad hoc gradient search method, has been the preferred training algorithm in the neural network community. Ease of implementation, utilization of the inherent parallel structure, and the ability to work on large data sets are the main arguments justifying the use of this method. However, in the present case where the data sets are of limited size, back-propagation is not the best choice. Instead we have decided to use the so-called *Levenberg-Marquardt method* for solving the optimization, since like conjugate gradient approaches it is in many ways superior to back-propagation as well as most other gradient search methods. The Levenberg-Marquardt method, independent of its neural implementation, is a work horse of many optimization packages (ref. 5). Some important advantages of the method are speed, guaranteed convergence to a (local) minima, numerical robustness, and minimal user-specified inputs are necessary except for providing a network architecture. Moreover, as pointed out in Moré (ref. 6) the method is surprisingly free of ad hoc solutions to achieve these benefits. Such advantages are important properties in making a user-friendly, easy-to-apply tool, which is crucial in this case since our objective was to create a generic methodology for application use and determine if in fact neural networks were capable of performing the complex mappings required in nonlinear aero design.

The Levenberg-Marquardt method has numerous variations. The simplest strategy may be found in the original contribution of Marquardt, while one adaptation to neural network training is discussed in reference 7. The version used here belongs to the class of *trust region methods* found in Fletcher (ref. 8). Just as back propagation, the Levenberg-Marquardt algorithm is an iterative search scheme

$$\theta^{(k+1)} = \theta^{(k)} + \mu^{(k)}h^{(k)} \tag{9}$$

From the current iterate $\theta^{(k)}$, a new iterate is found by moving a step of size $\mu^{(k)}$ in direction $h^{(k)}$. There exist several methods that fit into this structure. Their differences lay mainly in the way that a search direction is determined. In back propagation, a search direction is chosen as the gradient of the cost function evaluated at the current iterate

$$h^{(k)} = G(\theta^{(k)}) \equiv V'(\theta^{(k)}) = \left.\frac{\partial V(\theta)}{\partial \theta}\right|_{\theta=\theta^{(k)}} \tag{10}$$

while the step size can be either constant or vary according to some adaptive scheme. For a given cost function, the gradient is determined by

$$\boxed{G(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial\hat{\varepsilon}^i(\theta)}{\partial\theta}\varepsilon^i(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial\hat{y}^i(\theta)}{\partial\theta}\varepsilon^i(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\psi^i(\theta)\varepsilon^i(\theta)} \tag{11}$$

$\psi^i(\theta)$ denotes the gradient of the network output w.r.t. each of the weights, when the input is φ^i

$$\psi^i(\theta) = \frac{\partial \hat{y}^i(\theta)}{\partial \theta} \tag{12}$$

Another alternative method is the Gauss-Newton algorithm. Although convergence is not guaranteed and it suffers from severe numerical ill-conditioning, it provides an important basis for the Levenberg-Marquardt method. The idea is at each iteration to minimize a cost function based on a linear approximation of the prediction error

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}\left(\varepsilon^i(\theta^{(k)}) - \left(\psi^i(\theta^{(k)})\right)^T(\theta - \theta^{(k)})\right)^2 \tag{13}$$

It is shown (ref. 8) that this approach will fit into the basic scheme by setting the step size to 1, and the search direction to

$$h^{(k)} = \left[R(\theta^{(k)})\right]^{-1} G(\theta^{(k)}) \tag{14}$$

R represents the "Gauss-Newton" approximation to the Hessian ($V''(\theta)$) and is defined by

$$R(\theta) \equiv \frac{1}{N}\sum_{i=1}^{N}\psi^i(\theta)(\psi^i(\theta))^T \tag{15}$$

In the Levenberg-Marquardt method however, the search direction is found as an "intelligent" interpolation between the two previously mentioned directions as follows:

1) Create an initial parameter vector $\theta^{(0)}$ and an initial value $\lambda^{(0)}$

2) Determine the search direction, h, by solving the following system of equations
$$\left[R(\theta^{(k)}) + \lambda^{(k)}\right]h^{(k)} = -G(\theta^{(k)})$$

3) Evaluate network and determine $V(\theta^{(k)} + h^{(k)})$ as well as the "prediction" of the error $V(\theta^{(k)}) - L(\theta^{(k)} + h^{(k)})$

4) $V(\theta^{(k)}) - V(\theta^{(k)} + h^{(k)}) > 0.75\left[V(\theta^{(k)}) - L(\theta^{(k)} + h^{(k)})\right] \Rightarrow \quad \lambda^{(k+1)} = \frac{\lambda^{(k)}}{2}$

5) $V(\theta^{(k)}) - V(\theta^{(k)} + h^{(k)}) < 0.25\left[V(\theta^{(k)}) - L(\theta^{(k)} + h^{(k)})\right] \Rightarrow \quad \lambda^{(k+1)} = 2\lambda^{(k)}$

6) If $V_N(\theta^{(k)} + h^{(k)}) < V_N(\theta^{(k)})$ then accept $\theta^{(k+1)} = \theta^{(k)} + h^{(k)}$ as a new iterate

7) If the stop criterion is not satisfied set k=k+1 and go to 2). Otherwise set $\hat{\theta} = \theta^{(k)}$ and terminate (16)

Clearly, if λ is too large, the diagonal matrix will overwhelm the Hessian (R) in 2). The effect of this is a search direction approaching the gradient direction, but with a step size close to 0. This is important to ensure convergence, since the cost function always may be minimized by taking small

enough steps in the direction of the gradient. On the other hand, if λ equals zero, the method will coincide with the Gauss-Newton method. What then is a sensible strategy for adjustment of λ? The choice is basically to decrease λ if the approximation of error is reasonable and vice versa if it is not. This is just what is tested in steps 4) and 5) of the above algorithm. If a new iterate leads to a decrease in cost close to what is predicted using $L(\theta)$, λ is reduced. Since the right hand sides of 4) and 5) always are positive, λ always is increased until a decrease in cost is obtained.

From Madsen (ref. 9) it follows that

$$\left[V\left(\theta^{(k)}\right) - L\left(\theta^{(k)} + h^{(k)}\right)\right] = \frac{1}{2}\left(-\left(h^{(k)}\right)^T G\left(\theta^{(k)}\right) + \lambda^{(k)}\left\|h^{(k)}\right\|_2^2\right) \qquad (17)$$

which can be easily computed for use in step 3) and 4) of the procedure.

Some typical termination criteria for use in step 7) are:

a. The gradient is sufficiently close to zero.

b. The cost function is below a certain value.

c. A maximum number of iterations is reached.

d. λ exceeds a certain value.

Since the cost function will often have a number of local minima, it is important to run the training algorithm multiple times, starting from different sets of initial weights. The set of trained weights that leads to the lowest minima is then chosen.

GENERALIZATION

Before applying the above algorithm, a few comments should be made regarding mean square error cost functions. Actually, the mean square error criterion is not really what we are most interested in minimizing in this particular problem. This is especially so because the measurements available are corrupted by noise. A far better measure of fit is the mean square error at "all possible" measurements. This is what is known as the generalization error, the ability to predict new measurements not seen during the training phase.

$$\overline{V}(\hat{\theta}) \equiv E\left\{V(\hat{\theta})\right\} \qquad (18)$$

Besides being unrealistic to compute in practice, this quantity does not give information about whether the selected network architecture is a good choice. $\hat{\theta}$ depends on the training set and is thereby a stochastic variable, which in turn means that $\overline{V}(\theta)$ is a stochastic variable. Taking the expectation of $\overline{V}(\hat{\theta})$ with respect to all possible training sets of size N, yields

$$J(M) \equiv E\left\{\overline{V}(\hat{\theta})\right\} \qquad (19)$$

which is called the *average generalization error* or *model quality measure* (ref. 10). Assuming the existence of set of "true" weights, θ_0, allowing us to exactly describe the data-generating function by

the network architecture ($g(\varphi;\theta_0)=g_0(\varphi)$), and assuming the noise contribution is white noise independent of the inputs, an estimate of the average generalization error may be achieved. This estimate is known as Akaike's final prediction error (FPE) estimate. See ref. 10 for a derivation.

$$\boxed{\hat{J}(M)=\frac{N+d_M}{N-d_M}V(\hat{\theta})} \tag{20}$$

d_M is the total number of weights in the network, while N as before is the size of the training set.

The more we increase the size of the network, the more flexibility we add and the better we are able to fit the training data. In other words, $V(\theta)$ is a decreasing function of the number of weights in the network. If too much flexibility is added, one may expect that both the essential properties of the training set are captured and unfortunately, the properties of the particular noise sequence present in the data. This is commonly known as over fitting, and is exactly the dilemma Akaike's FPE expresses. There exist two usual approaches to deal with this problem. One is to find an 'optimal' network architecture (an architecture that minimizes J). The most successful strategy developed so far is pruning (see ref. 11). However, in order for it to be applicable, the data set should not be too limited compared to the required network size.

A second approach is to introduce a simple yet powerful extension to the cost function, called Regularization (or weight decay). Given the previously mentioned assumptions, it is a known result that the least squares estimate is unbiased. But unfortunately

$$E\left\{\left\|\hat{\theta}\right\|_2^2\right\}=\|\theta\|_2^2+\frac{\sigma^2}{N}tr\,(R^{-1}) \tag{21}$$

In other words, when minimizing the mean square error the weight estimates tend to be exaggerated. Imposing a punishment for this tendency in the cost function is called (simple) Regularization

$$W(\theta)=\frac{1}{2N}\sum_{i=1}^{N}\varepsilon_i^2(\theta)+\frac{\delta}{2N}\|\theta\|_2^2 \tag{22}$$

The simple Regularization approach leads to biased weights, since the weights are pulled towards 0. But by choosing the scalar sufficiently small, it can be shown that the average generalization error will decrease. See Sjöberg and Ljung (ref. 12) for a proof. Finding the optimal value is not a trivial task. More detailed discussions may be found in references 13 and 14. A rule of thumb is that a little Regularization usually helps. Other nice properties about Regularization are that it significantly reduces the number of local minima, as well as the number of required iterations to find a minimum. The Regularization extension to the cost function clearly influences the training algorithm, but incorporation is quite straightforward. In algorithm (16), we make the following changes:

The gradient of W becomes

$$G(\theta)\equiv W'(\theta)=-\frac{1}{N}\left(\sum_{i=1}^{N}\psi^i(\theta)\varepsilon^i(\theta)+\delta\theta\right) \tag{23}$$

Also the Hessian is changed, which in turn changes the expression for determination of the search direction

$$\left[R(\theta^{(k)}) + (\frac{\delta}{N} + \lambda^{(k)})I \right] h^{(k)} = G(\theta^{(k)}) \tag{24}$$

Since this expression from a numerical conditioning standpoint is the weak link of the training algorithm, it should be noticed that a spin-off from regularization is a robustness increasing effect on training, since the matrix (the term in brackets) will be moved further away from singularity. This is performed as follows:

Substitute V for W in steps 3 to 6 of (16).

Also L is changed to:

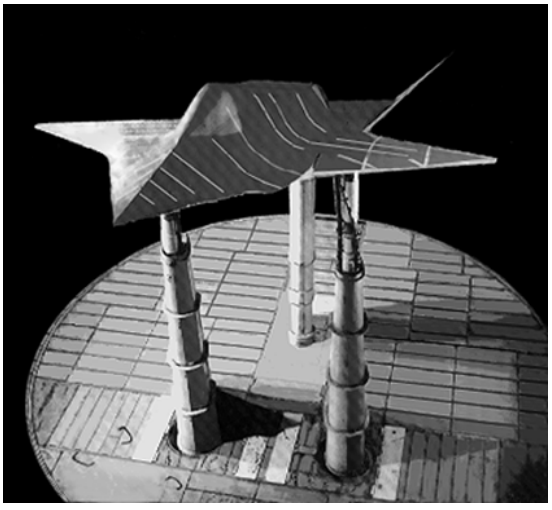
$$\overline{L}(\theta) = L(\theta) + \frac{\delta}{2N} \|\theta\|_2^2 \tag{25}$$

leading to

$$\left[V(\theta^{(k)}) - \overline{L}(\theta^{(k)} + h^{(k)}) \right] = \frac{1}{2} \left((h^{(k)})^T G(\theta^{(k)}) + \frac{(\lambda^{(k)} + \delta)}{N} \|h^{(k)}\|_2^2 \right) \tag{26}$$

APPLYING NETWORKS TO THE MEASUREMENTS

The aircraft used for method validation was a 55% scale model of the SHARC aircraft. See Picture 1.



Picture 1: SHARC aircraft mounted in the 40×80 ft. wind tunnel.

The test was conducted in the 40×80 ft. wind tunnel at NASA Ames Research Center part of the National Full-Scale Aerodynamics Complex shown in Picture 2.

Picture 2: National Full-Scale Aerodynamics Complex.

The control surfaces used in this study included leading edge and trailing edge flaps. During tests the left and right controllers for each flap were coupled. Each flap has six different set angles, thus thirty-six flap combinations were possible. The test involved picking a flap combination and varying angle of attack over a prespecified range, while measuring lift, drag, and pitching moment. Twenty-three of the total of thirty-six possible flap combinations were examined in the test, and the measurements were taken at sixteen different angles of attack (alpha) values for each combination. Table 1 presents the organization of the full data set and how the twenty-three subsets were separated into training and test sets, respectively.

Table 1. Organization of the data set for the 10% scale model test. "Train" indicates that the subset was included in the training set, similarly for "test". The entire data set consists of measurements from 348 different alpha/LE/TE combinations.

LE/TE	0.000	0.167	0.333	0.500	0.667	1.000
0.000	Train		Train		Train	Train
0.333	Train		Train		Train	Train
0.500	Test	Test	Train	Test		
0.667	Train	Test	Train		Train	Train
0.833	Test	Test				
1.000	Train		Train		Train	Train

Despite the possible differences in complexity, four identical network architectures were chosen to predict C_L, C_D, C_M and L/D.

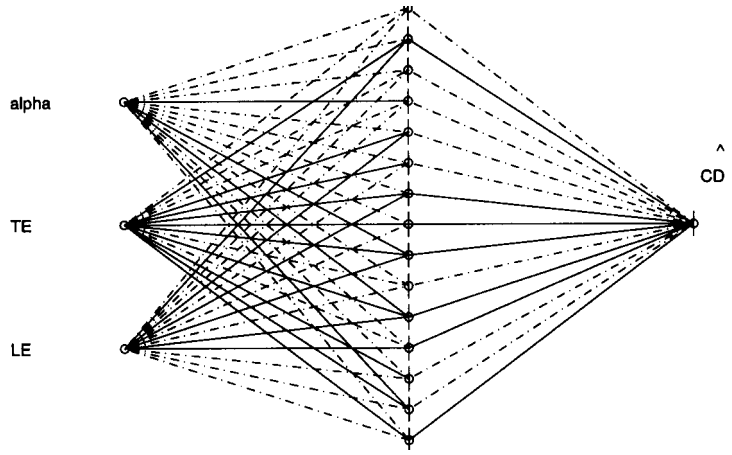


Figure 2. Network trained to predict C_D. All networks have one hidden layer containing 15 tanh units.

Training was carried out following the steps in the training phase process with a regularization parameter of $\delta = 10^{-3}$ in all cases. Inputs and outputs in the training set were scaled to unit variance. A typical learning curve is shown in figure 3. On an SGI Extreme II, the 200 iterations took about 40 seconds, but as can be seen from the figure, stopping earlier did not affect the final result significantly.

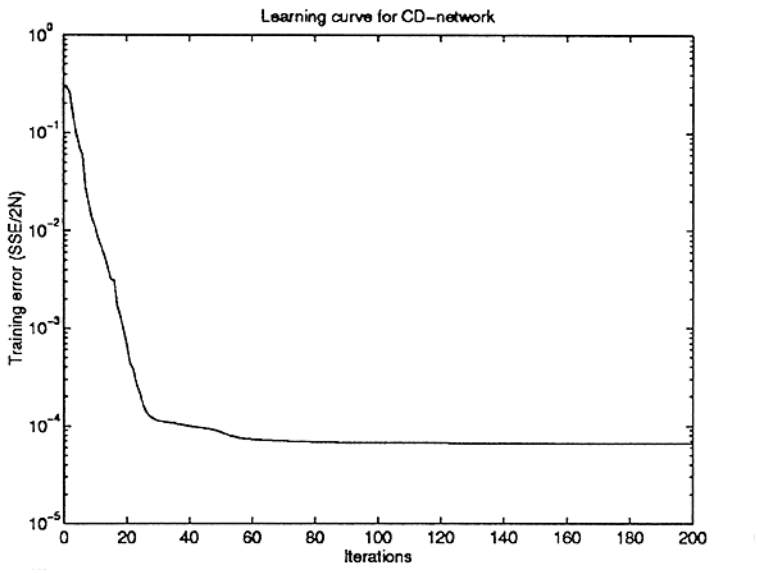


Figure 3. Learning curve (value of cost function as function of iteration) for network trained to predict C_D.

A number of test samples were available that could be used to help determine good network architectures (and regularization parameters) using cross validation. Due to the very high expenses associated with wind tunnel operations, this is unfortunately not the case in general. The reason for having a number of samples set aside in this case was that an important part of the study was to investigate the potential for reducing the number of test measurements. In another applied situation,

one should test different architectures (or vary the regularization parameter) and pick one that leads to a good compromise between having predictions close to the actual measurements and having the intermediate predictions following smooth curves. Typical results obtained from applying the trained networks to one of the six test sets are shown in figure 4.

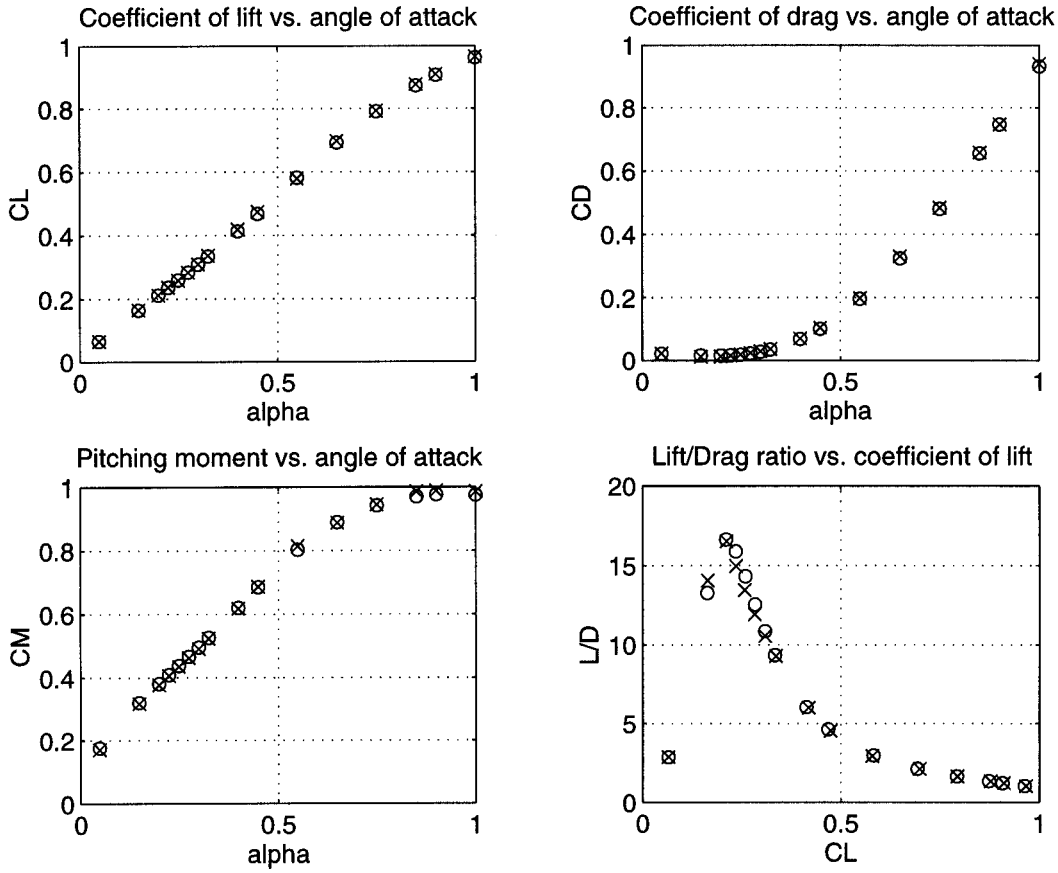


Figure 4**.** Comparison of test data and network predictions for LE=0.50 and TE=0.0.

It is difficult to come up with a good validity criterion in terms of an RMS value or a similar measure for this problem. Basically the validation was done by visual inspection of plots like those of figure 4. The predictions appear to be very close to the actual measurements, and taking the uncertainty of the measurements into consideration, the predictions are definitely considered to be satisfactory. L/D is harder to model than the 3 aerodynamical coefficients. An alternative way of predicting this ratio is to divide the predictions of C_L and C_D

$$\hat{L/D} = \hat{C}_L / \hat{C}_D \tag{27}$$

Unfortunately this strategy is very sensitive to prediction errors for small values of C_D, and compared to training on L/D directly, the performance was very poor. Notice that L/D is plotted versus C_L in figure 4. The reason why this figure is of particular interest will be explained in the following section.

ADVANCED APPLICATIONS

Because the neural networks have provided models that capture the relations between inputs and outputs, other utilization beyond new point estimation is straightforward. Two applications are considered here, both dealing with the problem of finding flap settings that ensure high maneuverability.

Overall flap setting

If the plane is flown with minimal change in flap settings, it is desirable to keep the flaps in a position that will ensure a high L/D over the topical flight envelope. In the current case this is interpreted in terms of a performance index we want maximized. The criterion is the area below the L/D vs. C_L curve in the C_L range [0.15, 0.55], as illustrated in figure 5.

$$J(LE,TE) = \int_{0.15}^{0.55} L/D(C_L, LE, TE)dC_L \tag{28}$$

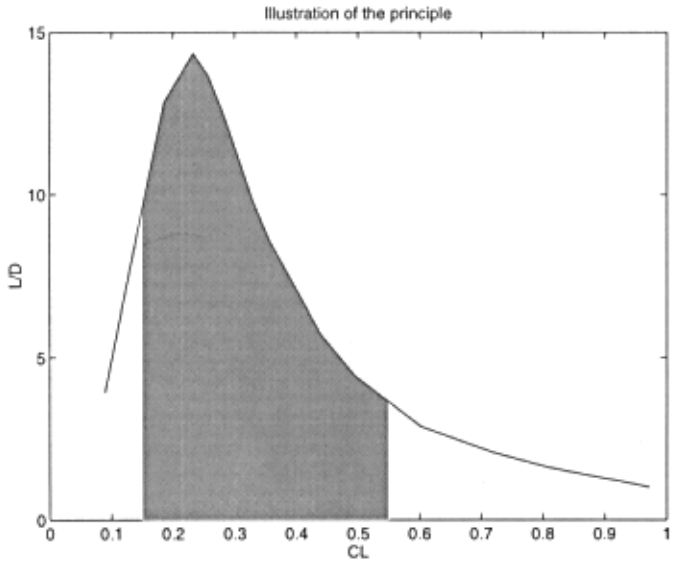


Figure 5. A rough sketch of the principle.

Basically the entire surface (J) is useful, but we are particularly interested in the maximum point

$$\overline{J} = \underset{\{LE,TE\}}{arg\max}\, J(LE, TE) \tag{29}$$

To find the areas (J), we need a way to express the L/D ratio as a function of C_L. To obtain this, a network is trained as the "inverse" function

$$\hat{\alpha} = \hat{C}_L^{-1}(C_L, LE, TE) \tag{30}$$

Since for fixed flap positions, C_L vs. α is an almost straight line over the necessary range of α, modeling the inverse function is not harder than modeling the actual function. In general, C_L vs. α

need not be one-to-one in the measured range, and if that is the case, one has to be extra careful about the training. The inverse function network should only be trained in the α range where the function actually is one-to-one. By using this new network in front of the L/D-network, we get a predictor for L/D which depends on LE, TE, and C_L.

Alternatively, a network L/D (C_L,LE, TE) might be trained directly, but for some reason this didn't seem to give quite as good results. The performance criterion (J) was then evaluated for a large number of flap combinations by applying numerical integration. The integration was carried out using the trapezoidal rule

$$\int_{X_1}^{X_N} f(x)dx \approx h\left[\tfrac{1}{2} f(x_1) + f(x_2) + \cdots + f(x_{N-1}) + f(x_N)\right] \qquad (31)$$

J was evaluated at 961 points (each flap was set in 31 different positions within their respective ranges). For each combination L/D was evaluated at 101 different values of C_L in the interval [0.15;0.55]. The result is shown below (figs. 6 and 7), using three different representations.

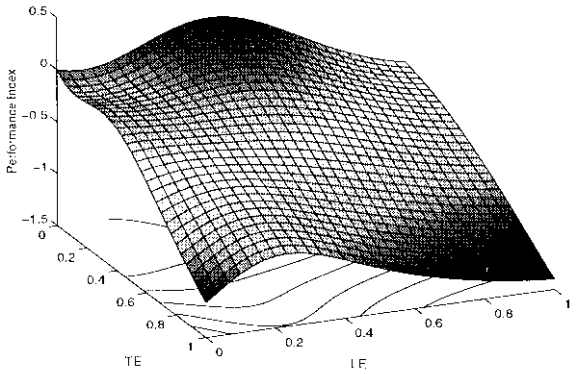


Figure 6. Surface plot of the integrated L/D for flap combinations of the 55% model.

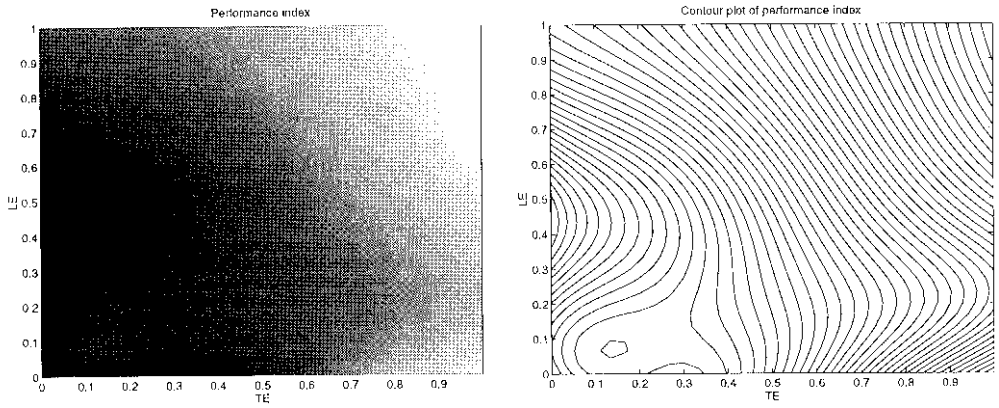


Figure 7. (a) The surface plot projected onto the TE-LE plane. (b) As a contour plot.

Flap schedules

To ensure a high maneuverability at all times (i.e., for all angles of attack) it is desirable to constantly position the flaps so that maximum L/D is obtained. Doing this is straightforward. For a particular choice of alpha, the L/D-network is evaluated at a number of different flap combinations, and the combination leading to maximum L/D is stored. This is then repeated for a large number of alpha values. The result of this procedure is shown in figure 8.

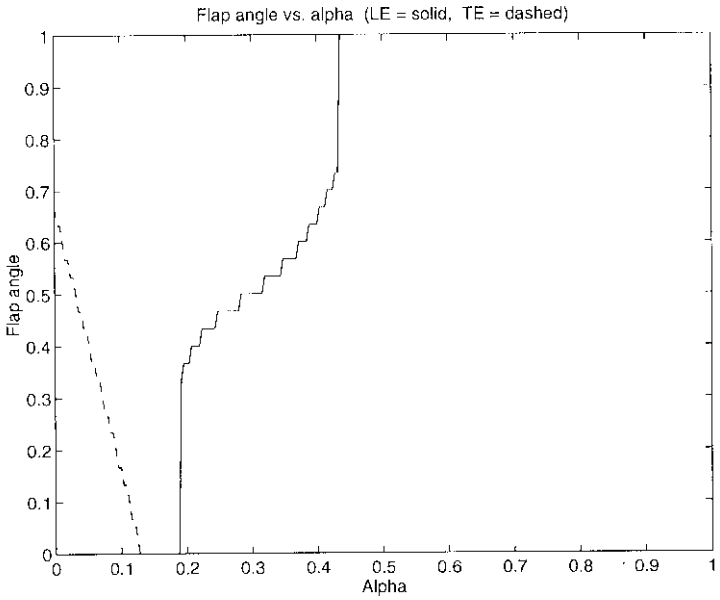


Figure 8. The schedule for each of the two flaps. For 101 different values of angle of attack maximum L/D was searched for among 31*31 flap combinations.

CONCLUSIONS AND FUTURE DIRECTIONS

The purpose of this study was threefold: To test the neural net method, to produce cost savings by minimizing the number of required wind tunnel measurements, and to automate follow-on data processing. We are very encouraged by the results obtained so far. In general, the predictions provided by networks trained on less than 74% of the original set of measurements are considered to lay within the tolerance ranges. Moreover, the peak in figure 6 matched closely the one found by the conventional procedure. In December 1994, the tool was applied during test of a newly generated 50% scale model of the SHARC. Again with impressive results, indicating approximately 40% savings (in this case certain alpha vales instead of certain flap combinations were discarded). A substantial spin-off from the study has been to provide plots like the ones shown in figures 6–8. By following conventional procedures, generating this type of plot is seldom realistic from a cost and resource viewpoint. The current method is very rapid and extremely cost effective for this task.

We believe that neural networks will become an important tool in future NASA Ames efforts to move directly from wind tunnel tests to virtual reality simulations of actual full scale aircraft flight behavior. Being able to fly the plane at such an early stage would be a tremendous help to flight engineers. Many bugs could be eliminated at a much earlier stage, which in turn would significantly

cut expenses, as well as time spent on development. Preliminary simulations in our laboratory have already demonstrated the principle. Efforts are now underway to explore the application of this technique to the estimation of hypersonic flight performance.

REFERENCES

1. Hertz, J.; Krogh, A.; and Palmer, R.G.: "Introduction to the Theory of Neural Computation," Addison-Wesley, 1991.

2. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math Control Signal Systems, 2, 303-314, 1989.

3. Haykin, Simon S.: "Neural Networks: a comprehensive foundation," Maxwell Macmillan International, 1994.

4. Sjöberg, J.; Hjalmerson, H.; and Ljung, L.: "Neural Networks in System Identification." Proceedings 10th IFAC Symp. on System Identification, Copenhagen, Vol. 2, pp. 49-71, 1994.

5. MathWorks (1994a): "MATLAB - Users Guide." MathWorks (1993): "Optimization toolbox for MATLAB." MathWorks (1994b) "Neural Networks Toolbox for MATLAB," The MathWorks, Inc.

6. Moré, J.J. : "Recent Developments in Algorithms and Software for Trust-Region Methods," Technical Memorandum No. 2, Argonne National Laboratory, 1982.

7. Hagan, M.T.; and Menhaj, M.B.: "Training Feedforward Networks with the Marquardt Algorithm," IEEE Transactions on Neural Networks, Vol. 5, No. 6, Nov. 1994, pp. 989-993.

8. Fletcher, R.: "Practical Methods of Optimization," Wiley and Sons, 1987.

9. Madsen, K.: "Lecture Notes on Optimization" (in danish), Institute for Mathematical Modeling, DTU, 1991.

10. Ljung, L.: "System Identification, Theory for the User," Prentice-Hall, 1987.

11. Hassibi, B.; Stork, D.G.; and Wolff, G.J. : "Optimal Brain Surgeon and General Network Pruning," in Proc. of the 1993 IEEE Int. Conference on Neural Networks, San Francisco, pp. 293-299, 1993.

12. Sjöberg, J.; and Ljung, L. : "Overtraining, Regularization, and Searching for Minimum in Neural Networks." Preprints of the 4th IFAC Int. Symp. on Adaptive Systems in Control and Signal Processing, pp. 669-674, July 1992.

13. Larsen, J.; and Hansen, L.K.: “Generalization Performance of Regularized Neural Network Models,” Proc. of the 1994 IEEE Neural Networks in Signal Processing Workshop, Greece, 1994.

14. Hansen, L.K.; Rasmussen, C.E.; Svarer, C.; and Larsen, J.: “Adaptive Regularization.” Proc. of the 1994 IEEE Neural Networks in Signal Processing Workshop, Greece, 1994.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1997	Technical Memorandum

4. TITLE AND SUBTITLE

Neural Network Prediction of New Aircraft Design Coefficients

5. FUNDING NUMBERS

519-30-12

6. AUTHOR(S)

*Magnus Norgaard, Charles C. Jorgensen, James C. Ross

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Ames Research Center, Moffett Field, CA 94035-1000 and
*Institute of Automation, Technical University of Denmark, Bygn. 326, DTU, 2800 Lyngby, Denmark

8. PERFORMING ORGANIZATION REPORT NUMBER

A-976719

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-112197

11. SUPPLEMENTARY NOTES

Point of Contact: Charles C. Jorgensen, Ames Research Center, MS 269-1, Moffett Field, CA 94035-1000 (415) 604-6725

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified-Unlimited
Subject Category - 01

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper discusses a neural network tool for more effective aircraft design evaluations during wind tunnel tests. Using a hybrid neural network optimization method, we have produced fast and reliable predictions of aerodynamical coefficients, found optimal flap settings, and flap schedules. For validation, the tool was tested on a 55% scale model of the USAF/NASA Subsonic High Alpha Research Concept aircraft (SHARC). Four different networks were trained to predict coefficients of lift, drag, moment of inertia, and lift drag ratio (C_L, C_D, C_M and L/D) from angle of attack and flap settings. The latter network was then used to determine an overall optimal flap setting and for finding optimal flap schedules.

14. SUBJECT TERMS

Neural networks, Wind tunnels, Coefficient estimation

15. NUMBER OF PAGES

20

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		